Saken Kenzhegulov
20132027

Machine Problem #1 (CSE351)
Report

The logic of my program is almost the same as the one of example code. Here's some changes I made to solve a given problem:

1) *myheader.h*
   a. Added <time> library to work with time functions.
   b. Instead of sending just string to server I send "Frame" structure that I've created to store all required information.

```
//structure of packet frame: seqno, client and server times, packet size, string
struct Frame {
        unsigned short seqno;
        struct timespec client_t;
        struct timespec server_t;
        unsigned short pkt_size;
        char string[81];
};
```

2) *echoServer.c*
   a. Added additional arguments and stored them in corresponding variables

```
if(argc != 4)
{
        printf("Usage: server <port(10000~30000)> <fixed delay(1~1000ms)> <bandwidth(1~1000Kbps)>\n");
        exit(1);
}
//get server port, fixed delay, bandwidth values
servPort = atoi(argv[1]);
fixDelay = atoi(argv[2]);
bandWidth = atoi(argv[3]);

if(servPort < 10000 || servPort > 30000 || fixDelay < 1 || fixDelay > 1000 || bandWidth < 1 || bandWidth > 1000)
{
        printf("Usage: server <port(10000~30000)> <fixed delay(1~1000ms)> <bandwidth(1~1000Kbps)>\n");
        exit(1);
}
```

   b. Implemented all steps from receiving a frame, and counting a delay, to a sending the frame back to client

```
while(1) {
        /* send Echo */
        clientAddrLen = sizeof(clientAddr);
        len = recvfrom(ls, &msg, sizeof(msg), 0,
                (struct sockaddr*) &clientAddr, (socklen_t *) &clientAddrLen);
        //print received packet frame contents
        printf("+ sent at: %lld.%.9ld / seqno: %d / size: %d / %s\n", (long long)msg.client_t.tv_sec, msg.client_t.tv_nsec, msg.seqno, msg.pkt_size, msg.string);
        //calculate delay time
        double wait =(fixDelay/1000.0+(8.0*msg.pkt_size)/(bandWidth*1000.0));
        struct timespec wait_t;
                wait_t.tv_sec = 0;
                wait_t.tv_nsec = (long long)(wait*1000000000);
        //wait (or sleep)
        printf("Sleep for %f second\n", wait);
        nanosleep(&wait_t, NULL);
        //obtain current server time and update received frame
        if( clock_gettime( CLOCK_REALTIME, &msg.server_t) == -1 ) {
                printf("clock gettime\n");
                exit(1);
        }
        // send back the frame
        n = sendto(ls, &msg, sizeof(msg), 0,
                (struct sockaddr *)&clientAddr, sizeof(clientAddr));
        if(n < 0) {
                perror("Error:\n");
                exit(1);
        }
        msg.string[strlen(msg.string)] = '\0';
        //print contents of sent frame
        printf("- echo at: %lld.%.9ld / seqno: %d / size: %d / %s\n", (long long)msg.server_t.tv_sec, msg.server_t.tv_nsec, msg.seqno, msg.pkt_size, msg.string);
        // check sentinel
        if(strcmp(msg.string, ".") == 0) {
                close(ls);
                break;
        }
}
```

3) *echoClient.c*

a. Added additional arguments and stored them in corresponding variables

```
if(argc != 5)
{
        printf("Usage: client <server IP> <server port> <pkt_size> <string>\n");
        exit(1);
}
//get server IP, Port, packet size, string values
servName = argv[1];
servPort = atoi(argv[2]);
```

b. Implemented all steps as getting and printing current time, constructing a frame, and as sending and receiving the frame from the server. Then printed received server time and string

```
//obtain the current client time and construct frame client time
if( clock_gettime( CLOCK_REALTIME, &msg.client_t) == -1 ) {
      printf( "clock gettime\n" );
      exit(1);
}

//print current client time
printf("TX at %lld.%.9ld\n", (long long)msg.client_t.tv_sec, msg.client_t.tv_nsec);

//construct frame's additional info
msg.pkt_size = atoi(argv[3]);
msg.seqno = 0;
char *string = argv[4];
printf("%d\n", n);
strncpy(msg.string, string, n);
msg.string[n] = '\0';

// send the frame
n = sendto(s, &msg, sizeof(msg), 0,
        (struct sockaddr *)&serverAddr, sizeof(serverAddr));

// wait for the echoed frame
len = recvfrom(s, &msg, sizeof(msg), 0, NULL, NULL);

//print echoed frame server time and string
printf("RX at %lld.%.9ld\n", (long long)msg.server_t.tv_sec, msg.server_t.tv_nsec);
printf("String: %s\n", msg.string);
```

Server and client were run from uni06 and uni07, respectively. Finally, I got these outputs:

- Client output:

```
[cs20132027@uni07 echoWait]$ echoClient 10.0.7.241 29999 1000 hello
TX at 1443097111.084163000
RX at 1443097111.610815000
String: hello
```

- Server output:

```
[cs20132027@uni06 echoWait]$ echoServer 29999 500 500
+ sent at: 1443097111.084163000 / seqno: 0 / size: 1000 / hello
Sleep for 0.516000 second
- echo at: 1443097111.610815000 / seqno: 0 / size: 1000 / hello
```